1 GUI

Figure 1 Widgets

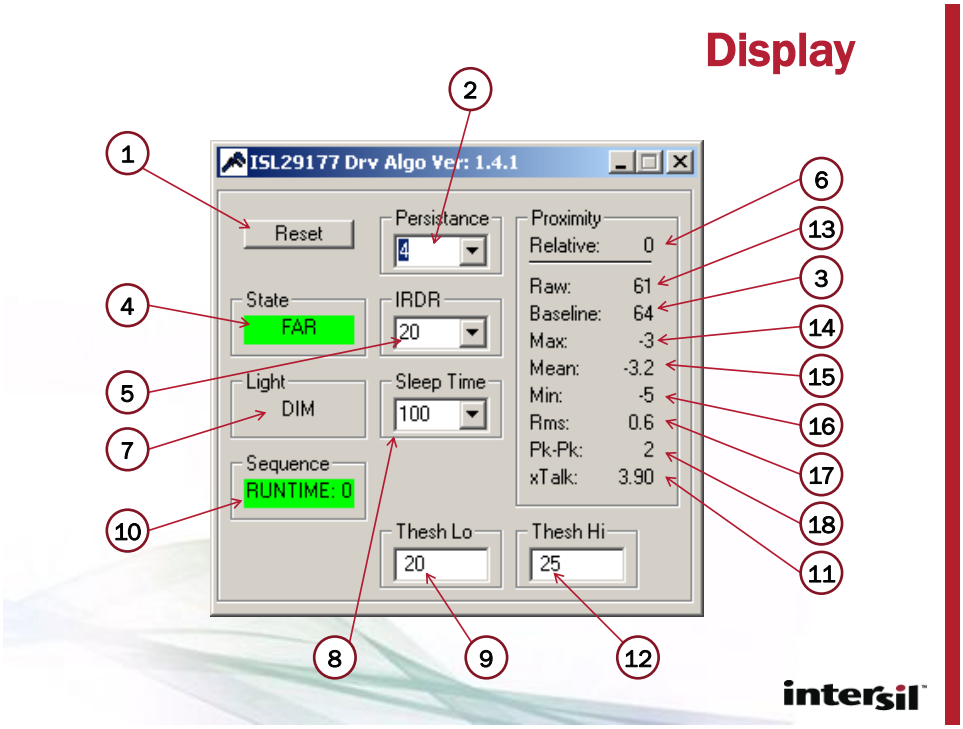
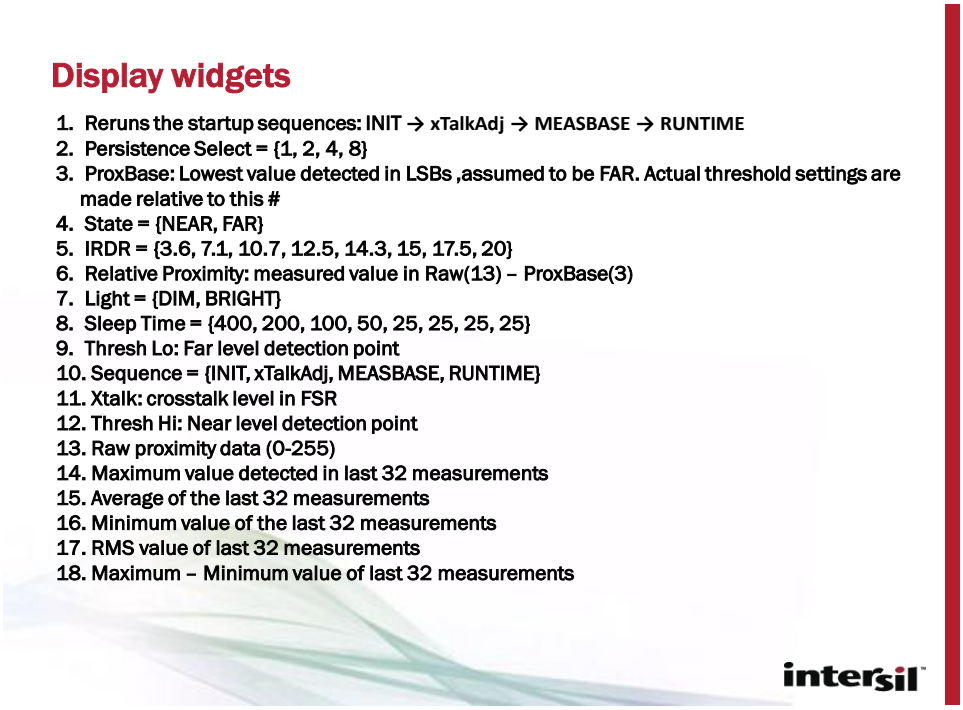


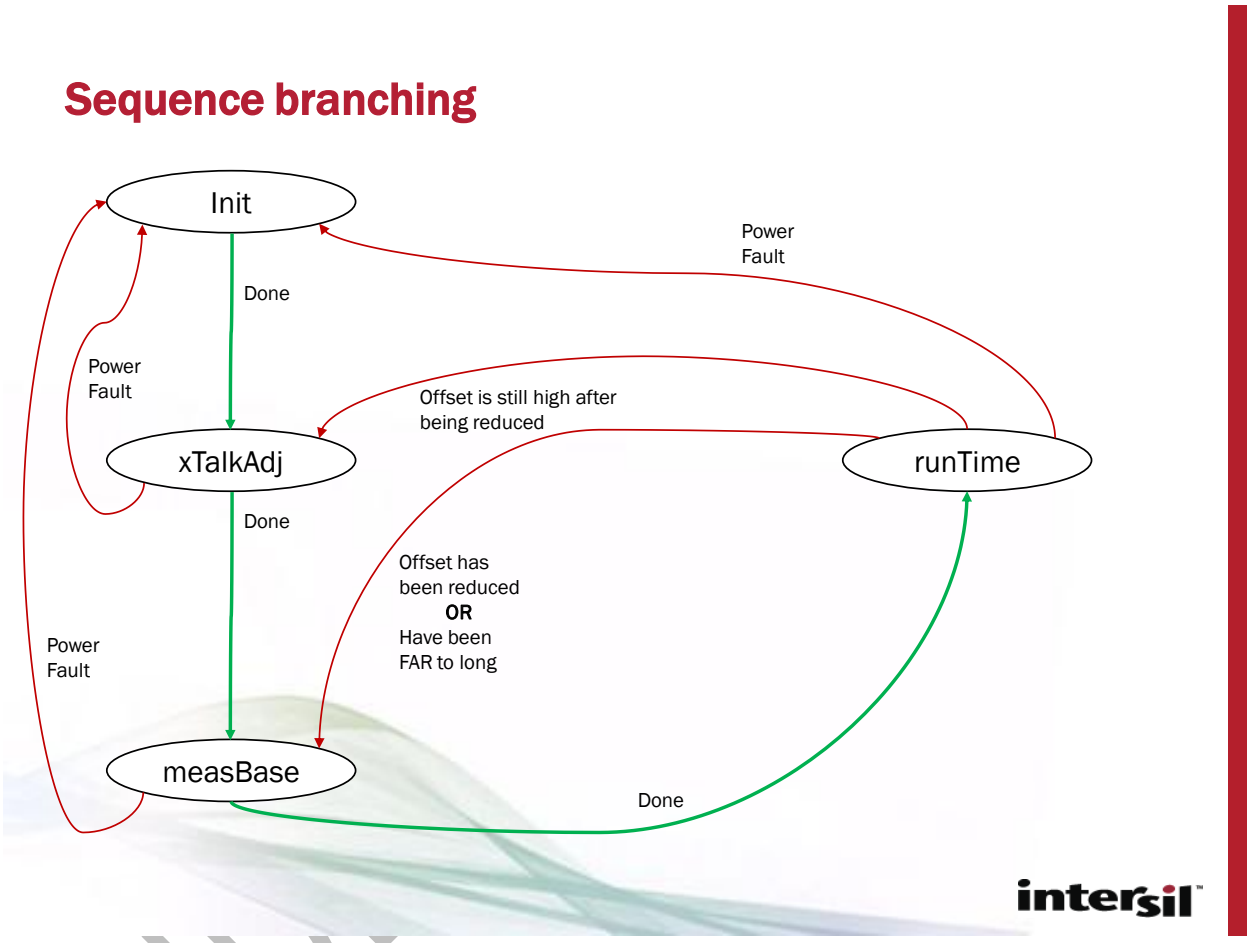
Figure 2 Widget Description



2 Sequences

2.1 Sequence Branching

Figure 3 Sequence Branching

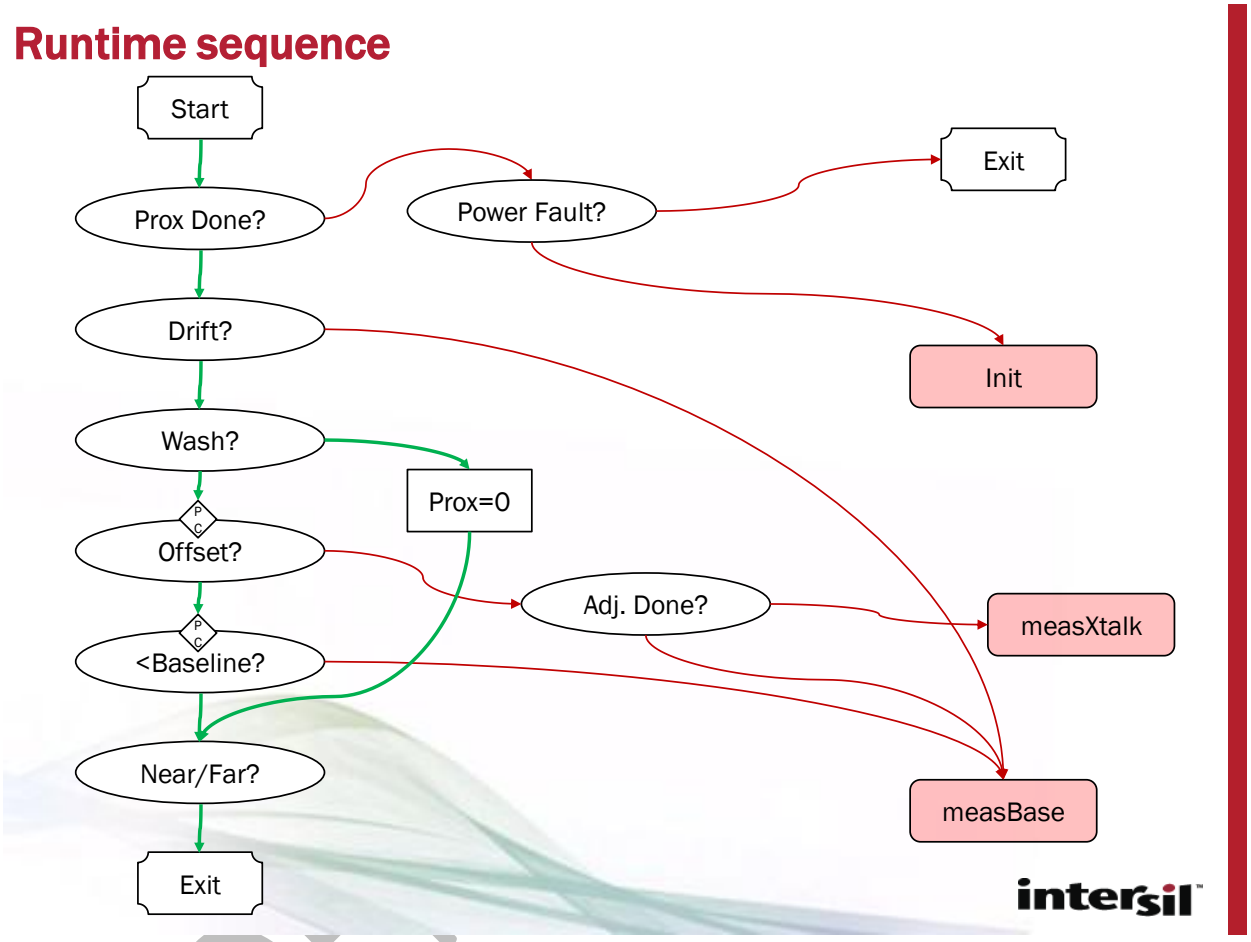


There are 4 sequences shown in Figure 3, Init, xTalkAdj, measBase and runTime. Each of these is driven by timers. Only one is active at a time (baton passing). Upon successful completion they branch to the next item until runTime is operational. This path is shown in **green** in the figure. Each of them may need to branch to a previous item if any fault conditions are detected. The branch back paths are shown in **red**.

The Init should only be run once on 1st startup. If a power fault is detected during operation it should be used to reset the device.

2.2 Sequence: runTime

Figure 4 Sequence: runTime



The runTime sequence is the most important and complicated of the group of sequences. A "flow chart" of this sequence is shown in Figure 4. This sequence, like the others is polled. Normal flow is from top to bottom on the left of the diagram, along the **green** paths. Branching to other sequences is along the **red** paths with the exit point also shown in red. The individual decision points are detailed below:

1. Prox Done? / Power Fault?

   Purpose: Checks status register

   Read *STATUS once* (it clears on read). Exit if the conversion is not complete. Reinitialize if the supply flag is set (power fault has occurred). Otherwise continue.

   a. **IF** [STATUS:PROX_DONE (0x6:B2) == 0] **EXIT**
   b. **ELSE**
      i. **IF** [STATUS:SUPPLY_FLAG (0x6:B4) == 1] **Init** // power fault
      ii. **ELSE** Drift?

2. Drift?

   Purpose compensates for long term drift.

   Once per *user defined* (default = 60 seconds) re-measure the baseline provided the proximity value is NEAR.

3. Wash?

   Purpose: prevent self rise, if IR is detected for proximity value to 0, FAR

   if PROX_WASH > minimum value detected +1 then proximity=0, go to Near/Far?
   if minimum value detected > PROX_WASH then minimum value detected = PROX_WASH

4. Offset?

   Purpose: Make sure minimum proximity detected > 0

   After *user defined* (default = 3 cycles) persistence of proximity < 0

   a. Decrement LUT index by 1 (once), go to measBase
   b. If decrement done (4.a) then go to measXtalk

5. <Baseline?

   Purpose: correct for incorrect Far level (baseline)

   After *user defined* (default = 3 cycles) persistence of proximity < baseline go to measBase

6. Near/Far?

   Purpose: return Near/Far result

   If proximity < threshLo return FAR
   If proximity > threshHi return NEAR
   Else return last state

Example 1. VB6 Code

```
Private Sub tmrRunTime_Timer()
    Dim prox As Double, wash As Byte, offset As Long, pFlag As Boolean, np As Byte, pram As Long
    Static nearFar As Byte, lastProx As Double, lastBase As Double, lPflag As Boolean

    Static offsetReduce As Boolean      ' done already, remeasure xTalk
    Static offsetReductionCount As Long

    Static baseLinePersistCount As Long
    Static driftCounter As Long
    '
    ' This routine contains 3 possible state changes
    '   1) Init: if "brown out" detected (initNeeded)
    '   2) measBase: if prox = 0 for consecutive offsetPersist cycles (offsetReductionCount)
    '   3) xTalkAdj: is offset has been adjusted already but is still 0

    Static primed As Boolean ' start on 2nd call

    If primed Then

        If als.GetProximity(prox) Then ' PROX_DONE

            prox = Int(prox * 255)

            ' ============================================================
            ' occasionally remeasure the baseline to compensate for drift
            ' ============================================================
            driftCounter = driftCounter + 1
            If driftCounter > 1000 * measBaseTime / (tmrRunTime.Interval) And prox < threshLo + baseline Then
                primed = False: gotoState state.stMeasBase
            End If

        Else

            ' =====================
            ' check for power fault
            ' =====================
            If als.getInitNeeded Then ' reset sequence (brown out)
                primed = False: gotoState state.stInit
            End If

            Exit Sub

        End If

        ' ================================================
        ' monitor the 2/3 pulse bit (for debug/testing only)
        als.readField BF.XPLS_, np
        lblSeq.caption = "RUNTIME:" & Str(np)
        ' ================================================

        pFlag = als.getPflag

        If pFlag <> lPflag Then
            lPflag = pFlag ' break here for flag change
        End If

        If pFlag Or 1 Then ' Poll/Interupt And:Use Interrupt; Or:Poll

                           ' XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            dataValid = True ' unless offset too high REMOVE
                           ' XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
' ============================
' washout detection/ tracking
'als.GetProximity prox: prox = Int(prox * 255)
als.readField BF.WASH_, wash
If washThresh > wash Then washThresh = wash ' new washout detect value
If wash > washThresh + 1 Then ' guardband threshold by 1 LSB
    prox = 0 ' bright light detected, force to zero
    nearFar = 0
    lblLight.caption = "BRIGHT"
' ============================
Else
    lblLight.caption = "DIM"

    ' ================================================================
    ' check if offset is too high, drop value by 1, remeasure baseline
    ' if has already been reduced once, readjust offset  [PC]
    If prox = 0 Then
        offsetReductionCount = offsetReductionCount + 1
    Else
        offsetReductionCount = 0
        offsetReduce = False
    End If

    If offsetReductionCount >= offsetPersist Then

        If offsetReduce Then            ' if drop by 1 didn't work, readjust offset
            primed = False: gotoState state.stXtalkAdj
        Else                            ' drop by offset by 1, remeasure baseline
            offsetReduce = True
            drv.dGetProxOffset pram 'returns LUT index in pram
            If pram > 0 Then pram = pram - 1
            drv.dSetProxOffset pram ' pram sets LUT index, returns offset in LSBs
            primed = False: gotoState state.stMeasBase
        End If
    ' ================================================================
    Else

        ' ============================
        ' check for baseline level [PC]
        prox = prox - baseline
        If prox < 0 Then
            baseLinePersistCount = baseLinePersistCount + 1
            prox = 0
            If baseLinePersistCount >= baseLinePersist Then
                baseLinePersistCount = 0
                primed = False: gotoState state.stMeasBase ' remeasure baseline
            Else
                baseLinePersistCount = 0
            End If
        End If
        ' ========================
```

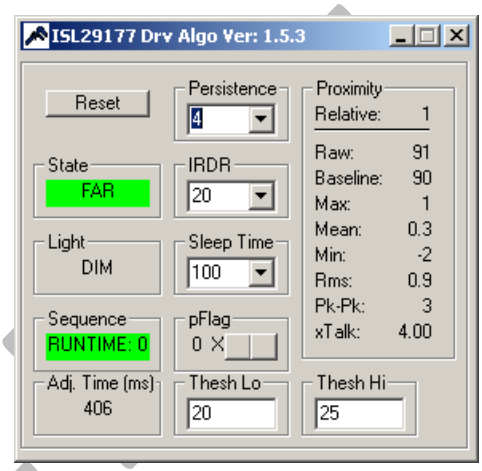
```
                    ' ==================
                    ' determine Near/Far
                    If 0 Then ' Near/Far 0: use prox value; 1: use Prox Int Flag
                        nearFar = pFlag
                    Else ' set to 1 for Motorola Logic

                        If prox > threshHi Then
                            nearFar = 1 ' Near
                        Else
                            If prox < threshLo Then
                                nearFar = 0 ' Far
                            Else
                                'in hysterisis band, use last value
                            End If
                        End If

                    End If
                    ' ==================

                End If

            End If ' wash/not

            updateProx prox, nearFar

        End If ' interrupt/poll loop

    Else
        driftCounter = 0
        primed = True
    End If

End Sub
```

3 Test Procedure

3.1 Functional (Black Box)

3.1.1 Initialization

Start Driver with no object present:



Observe for > 1 minute:

1. State = FAR
2. 10 < Baseline < 100
3. Proximity (relative) < 10
4. 10 ≤ Baseline ≤ 90

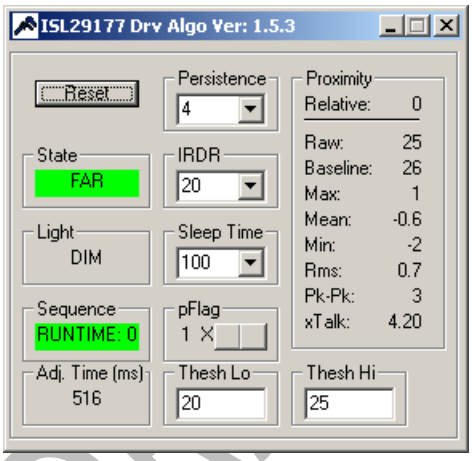
3.1.2 Near/Far

1. Place flat object (i.e. paper) ~3cm from sensor
2. Verify NEAR state: Proximity (relative) > Threshold High (default=25)
3. 10 ≤ Baseline ≤ 90

4. Remove object
5. Verify FAR state: Proximity (relative) < 10
6. 10 ≤ Baseline ≤ 90

3.1.3 False Offset recovery

1. Place flat object (i.e. paper) ~3cm from sensor
2. Reset driver
3. Verify FAR state: Proximity (relative) < 10
4. 10 ≤ Baseline ≤ 90



5. Remove Object
6. Verify FAR state: Proximity (relative) < 10
7. 10 ≤ Baseline ≤ 90

8. Return Object
9. Verify NEAR state: Proximity (relative) > Threshold
10. 10 ≤ Baseline ≤ 90